Open source technology in intraorganizational software development – Private markets or local libraries?

Juho Lindman

Aalto University School of Economics

Mikko Riepula

Aalto University School of Economics

Matti Rossi

Aalto University School of Economics

Pentti Marttiin

Aalto University School of Economics

Abstract. This chapter explores how two organizations have changed their software development practices by introducing Open Source technology. Our aim is to understand the institutional changes that are needed in and emerge from this process. This chapter develops a conceptualization building on the insights of entrepreneurial institutionalism, concentrating on the changing relationships of organizational groups in the areas of decision-making, rewarding and communication. We identify the links between the 1) emerging, yet embedded technology and 2) the underlying institutional decision-making, reward and communication structures. We move the Open Source 2.0 research agenda forward by concentrating empirical work on the nuances of institutional change that open source brings about in large hierarchical organisations. We will discuss the appropriateness of internal accounting organized according to the principle of an open market vs. a local library. We believe that both of these metaphors can support innovation, but different groups will find different approaches more appealing.
Keywords: Open Source, Entrepreneurial Institutionalism, Organizational Change.

Introduction

The topic of this chapter is how *open innovation technology,* rather than *open innovation* as such, changes an organisation. We study the institutional transformation caused by the introduction of Open Source Software (OSS) technology (practices and tools) within traditional software development organizations.[1] OSS literature often assumes a "bazaar" of development in a virtual organization characterized by loose control, openness and community orientation. However, inside a single large organization, where contributions came from employees or subcontractors, the setting is different. The companies introduce OSS practises and foster the creation of communities to serve their business needs, that is, to create quality products. Such arrangements often imply a looser structure, more open documentation, feedback from the user community and the introduction of agile practises. These development arguments are corroborated by business arguments of partial outsourcing to the developer community, cost savings from using common (sometimes external OSS) platforms and the possibility of creating industry standards through a wide availability of the finished products.

The identified phenomenon is important because open source technologies 1) are adopted in large organizations based only on a partial understanding of the nature of the institutional change they enable, drive, or necessitate, and 2) are not adopted in organizations because their consequences are seen to include unnecessary or unknown risks. We believe that building a conceptualisation based on extensive fieldwork will enable a better evaluation of these technologies and their contextual appropriateness.

Therefore our research questions are:

- How can the introduction of open innovation technologies, such as OSS technologies, be leveraged to improve development practises?
- What are the institutional effects of these changes?

---

1 We use the terms "OSS-style development" and "OSS practices" synonymously, encompassing "OSS technologies" as a form of open innovation technologies. Our main interest is how these can be used within companies developing products, not necessarily OSS as such. By "OSS technologies" we do not mean the license of the developed software, but the common infrastructural tools used in OSS communities. The tools include concurrent versioning systems, issue trackers, email-driven and archived communication, and web presence, which all support software development practices similar to OSS in creative commons, but in our cases within a single organization.

To answer these questions, we analyse two cases of OSS technology being introduced within a large corporation. Our goal is to build a conceptualization of what happens in a hierarchical systems development organization when OSS technology is adopted[2]. Informed by institutional theory (Scott, 2001; Greenwood and Hinings, 1996), we seek to identify the inertia caused by old institutional forces and the changes in reward structure and developer and manager mindset needed to realize the benefits of more open development

This chapter is organised as follows. In the second section we review relevant literature on OSS technology in commercial organizations. In the third section we develop a conceptualization to explain the transformation. The fourth chapter is about the research approach used. Case findings then demonstrate the links between the embedded technology decision-making and communication and reward structures. In the final section we conclude how OSS technology is leveraged in the case companies' systems development and what the accompanying institutional changes are.

OSS technologies in commercial organizations

OSS technologies have been successfully implemented in different organizations and OSS-style development based on distributed and global practices has gained industrial credibility (Fitzgerald, 2006). OSS as such is used more and more as an integral part of all kinds of products. OSS development is often characterized by a modular software architecture, distributed global development teams, meritocracy, voluntarism, often elaborate decision making mechanisms, and the technical and legal openness of the code which enables code inspection, bug reporting, and maintenance (Fitzgerald, 2006). OSS as such is traditionally defined as software licensed under an OSI-approved software license (Välimäki, 2005). OSS practices are practices that emulate development in an OSS community (technical infrastructure enabling communication, reward structures, supporting work and knowledge transfer). OSS practices often include the use of email (and the archives thus available) as the primary communication tool, availability of the code from a source code repository via CVS (Concurrent Versioning System) or

2 One of the main reasons for companies to adopt OSS technology is their interest in improving software reuse. At the same time companies are adopting distributed and virtual teamwork practises and changing their software development processes from waterfall to iterative, thus adopting agile practises (about traditional, agile and open source practises in Barnett, 2004). These two changes favour the adoption of OSS tools, but failed to address the challenge of reuse.

similar, web presence (for example Sourceforge), and some kind of issue tracker. The main difference between traditional (closed source) and OSS development is that the latter can sustain non-commercial communities as the source code is available to all. The source code might belong to its developer or the community in a way that prevents traditional software license sales (Dahlander and Magnusson, 2005). However, the availability of the source code outside the organisation is not a prerequisite on implementing practices similar to OSS inside a company (for example, Fitzgerald, 2006).

Inner source (van der Linden et al., 2009; Lindman et al., 2008) and corporate source (Dinkelacker and Garg, 2003) as terms refer to OSS practices limited inside companies. Often the introduction of OSS-style development starts with these tools, but as "tools are not only tools" their productive application might require fundamental changes in software development (Sharma, 2002). Inside a large organization (Wesselius, 2008; Gurbani et al., 2010) or in a business-to-business setting (Fink, 2003) the fundamental differences between OSS and traditional software are smaller than inside small software companies. The license and corporate policies and processes define how software is acquired, procured, installed, used, maintained and discarded. Furthermore, company guidelines, contracts and/or licenses also define how software is developed, remuneration acquired and benefits divided (Välimäki, 2005).

In the first phase of OSS commercialisation companies were interested in ways to directly benefit from the revenue stream created by OSS (Rajala, 2006). OSS research has traditionally focused more on individual motivations of the developers and community-driven development than OSS in hierarchical organisations (Stol and Babar, 2009). Now in the second phase of OSS commercialization, the use of OSS-style development processes is gaining a foothold in large commercial organizations (Gurbani et al., 2010; Fitzgerald, 2006; Santos, 2008).

Conceptual framework

Organizations are struggling to balance the possibilities offered by OSS technology, but research efforts have only recently started to focus more on organizational issues in large hierarchical organizations. We draw on literature streams of institutional theory and focus on entrepreneurial institutionalism to understand the phenomenon in organizational context.

Institutional theory

Institutional theory views institutions as "*multifaceted, durable social structures, made up of symbolic elements, social activities, and material resources*" (Scott, 2001, p. 49). Institutional structures, such as reward and communication structures, are set up by regulative, normative and cultural elements or pillars (Scott, 2001). Institutional theory (Powell and Dimaggio, 1991) has been accommodated to explain change (Greenwood and Hinings, 1996), even though it has been criticized for mainly focusing on "convergence" (similarity). It should be noted that institutional theory is far from a monolithic tradition (for a more thorough discussion about "old" and "new" institutionalism, see Powell and Dimaggio, 1991; Greenwood and Hinings, 1996))

Institutional theory underlines the "relationship" between an organisation's normative context and the varying interests of the groups (stakeholders) within the organisation. Functionally different groups in organisations are not neutral towards each other, but instead the technical boundaries of the groups are reinforced cognitively (Greenwood and Hinings, 1996). Our conceptual framework draws on institutional theory (Scott, 2001) and social constructionism by analysing using the concept of an "organizing vision" (Swanson and Ramiller, 1997). There are tensions between the traditions of institutionalism and social constructionism, but as Scott (2001) notes "*choice [in organizations] is informed and constrained by the ways in which knowledge is constructed…*" We posit that while normally the actors and proponents of organizational change truly subscribe to OSS inspired values for the better, "the OSS spirit", they are also renegotiating the exact meaning of OSS to fit the organisational context. These negotiations can be understood better by analysing the term "OSS" as a justification for organisational change. The exact meaning of adapted OSS is renegotiated and implies changes in the allocation of resources and the division of work between units.

Entrepreneurial institutionalism

Research in institutionalism, which focuses on change, is called entrepreneurial institutionalism. It is a response to the call for institutional theory to focus more on agency and organizational change (Garud et al., 2007). Work on institutions has traditionally focused on continuity (Garud et al., 2007, p. 960). In contrast, work on entrepreneurship has focused on change. In institutional theory, this contrast of structure and agency has been identified as the paradox of embedded agency (DiMaggio and Powell, 1991). One solution to this paradox is to view

structures as platforms for change rather than constraints (Garud and Karnoe, 2003).

Any new technology is a change in status quo, with winners and losers. The meaning of organizational visions (Swanson and Ramiller, 1997) is renegotiated within the boundaries of a certain language community and draw on local discursive resources. OSS technology is an organizational tool that stresses local issues regarding software production in the context of a certain organization. OSS also provides ways of addressing these issues. It can be seen as a metaphor used in an organisation that is making sense of its changing business environment so that it is able to operate in it. OSS often offers a promise of a more agile development approach, more contribution, more open discussion and less hierarchy in software development. In short, it gives certain justifications, reasoning and opportunities to a decision-maker faced with difficult decisions concerning reorganization or introducing a new organizational innovation (Van de Ven, 1993).

We use the institutional entrepreneurship lens to identify how the meaning of OSS technology changed during implementation and how our two organizations evolved when OSS technology was institutionalised. We aim to provide insight on the process of OSS technology institutionalisation and the underlying changes. In order to explain the institutionalisation of OSS technology we focus on three structures within the companies: the reward, decision-making and communication structures. However, we do not claim that these would be easily separated entities.

We chose the different organizational groups to highlight their different interest and incentives in the process. The different selected groups (stakeholders) are 1) the technology provider unit (the central group), 2) the technology user unit (business unit), and 3) the developer/users.

Research approach

The nature of our research problem, human behaviour and interaction, led us to use a qualitative research approach (Seaman, 1999; Klein and Myers, 1999). We chose a case study approach and adopted the principles of interpretive case studies.

Practical Tip. When planning organizational changes, understanding the current situation makes transitions processes smoother. This is especially true when a specific technology related to innovations is being adapted. Identifying and mobilization of the different stakeholders requires on-site research of the different organizational groups involved.

As the main data collection method, we applied semi-structured thematic interviews. We interviewed two to three persons per case organisation in three occasions over two-year intervals to better capture the nuances of the changing organisation. We stopped interviewing after the 14th interview, because recent interviews did not convey additional information regarding the actual events. Research design can thus be considered longitudinal. The first interviews were gathered in 2006 and the second round of interviews was conducted in 2008. The final round took place in 2010-2011. Most interviews lasted about one hour.

The interviewed people represented three different organizational groups, one person from the service provider group, one from the service user group and—except for the last round—one from the developer/user group. We chose managerial respondents from the business and central groups to gain an understanding of the management rationale for introducing OSS technology. The developers were included to bring in the user viewpoint.

One of the researchers works in one of the case companies and was therefore able to provide access to the organization and, as a "native", reflect on the organizational context. We were very careful to eliminate any bias this connection might introduce to the setting. In addition, we used secondary data obtained in the course of the industry research project such as project descriptions, manuals, portal usage data, documentation and visits to the sites to familiarize ourselves with the setting.

In the first two rounds, we analysed the interviews by first recounting the organizational history and change as described by the respondents. We circulated the transcribed interviews back to the respondents, so they could correct the views should they have been misinterpreted. The last round mainly focused on what had changed since the previous rounds of interviews.

The systematic analyses were based on pattern matching recurring themes between different interviews and then categorizing the data according to the themes.

The themes we focused on were how the respondents talked about 1) instituting new technology, 2) changes in the communication media and the reward structures between units and individuals, and 3) changes in the different ways the respondents described their group involvement. The authors extracted all the instances where the respondents talked about the themes and reported the findings.

We classified the findings into three areas: 1) how OSS technology is renegotiated to fit the organizational context and how OSS infrastructural tools are used inside companies, 2) how the respondents saw the change between business

units and central unit, and 3) how the respondents described the reward, decision-making and communication structures as both a platform and driver of change.

Cases

The two cases were selected among the partner companies of the ITEA-COSI project, which also set the context and enabled access to the case companies. ITEA-COSI was a joint academic and industrial project focused on software commodification.

Philips Inner Source

The offering of Philips Healthcare (PH) consists of a wide variety of medical systems, for example X-ray technology, ultrasound, magnetic resonance and information management. The factory-preinstalled software is customised and configured, but not sold separately. PH normally maintains the software for 10 years, which often leads to a large installed base and makes large changes very complicated. PH is maintaining and developing a large software base including a set of software components reused in all business units.

Historically components were developed in a central software group (Wesselius, 2008). In this configuration it was difficult to manage the different development activities and unaligned roadmaps. Lack of required domain knowledge in the central group made asset reuse difficult.

To solve these two issues, the business units started to contribute to developing new software assets. This would enable the business unit with the best domain knowledge to develop the software and then contribute it to a shared portfolio. Business units would not have to wait for the central group to develop the (often rushed and high priority) asset. OSS technology (tools and practices) was introduced in PH to legitimate the change.

The division of work was based on the idea that the central group was responsible for the common platform and business units developed add-ons, customised and configured the software. Components are distributed via intranet, email, ftp and CD. Business units choose the components for use, customization and configuration. Different groups offer services to each other (for example support and maintenance) based on agreements between internal customers. Developed

software was also made available to other business units. One of the main benefits of a common platform is that it would avoid duplicate work and promote the reuse of software. Co-development activities with business units and central group were favoured in order to benefit from organizational learning.

There were also certain risks involved mainly dealing with the distributed setting. The central group would become more dependent on several business units at the same time. The overall quality would be more difficult to control, if business units would only make stand-alone add-ons. Business unit incentives were also un-aligned, as it seems that there is no guarantee that units would actually contribute back and not only use the resulting code. This applies also to the maintenance of the software asset and balancing the maintenance between business units. The scenario where one business unit is putting a lot of resources and effort on development and maintenance, but all the business units would use the outcome was considered problematic.

The communication plan was to be as explicit as possible and share information with all the interested parties. Co-development activities required informal discussions between developers, but broader issues were decided in formal settings such as steering groups and operational teams. There were also formal architect meetings and a monthly platform group meeting that all interested parties could participate in. Information was also posted on the intranet and PH mailing lists. A back channel of communication were the so called marketers, who were chosen, per business unit, to promote inner source and gather feedback in case of problems. Development work is somewhat controlled by steering groups and operational meetings, but development was mainly driven by business groups which need new functionality.

A couple of years ago, Philips was thinking of a new scheme for sharing the development costs. The old model was based on centralised component development and a component tax where the central group did not have profit targets (Wesselius, 2008). The central group performed maintenance of the components. The component tax levied from business units was based on component development and maintenance activities and on an agreed upon roadmap on a yearly basis. Based on the relative amount of component usage and the size of the unit's external sales, the estimated costs were then distributed among the business units. Users of old component versions paid more for maintenance to offset the burden of maintaining many old versions.

When moving to an inner source approach, the component tax model is not ideal since it does not promote contributing to the shared component base. A business unit that contributes a reusable component has to make an extra effort to make the component reusable. Business units have profit targets and investing re-

sources to make components reusable is conflicting with these targets. It wasn't clear which group was expected to perform maintenance for the contributed component or allocate the maintenance resources. If the contributing business unit has to do the maintenance, this will again add costs to the unit. However, making the central component group responsible for maintenance would require this group to build competences for maintaining software components developed by other groups. The central group would be enlarged and take away the domain experts from the business units.

Nokia iSource

Nokia is one of the leading mobile communications companies. It is a publicly held company with listings in five major exchanges and in 2004 (prior to the merger of its Network unit with Siemens to form Nokia Siemens Networks or NSN) its net sales totalled EUR 29.2 billion. We study the organizational changes from the viewpoint of technology adoption and focus on the role of the source code portal called iSource.

The idea to adopt collaborative development utilizing open source software practices was presented for Nokia in the early 2000s. It was encouraged by the positive experiences when adopting open source practises in a company context (Dinckelacker et al., 2002). The aim was to tackle the challenges of reuse and cost-effective re-development of software with multiple parties. These challenges are typical of centrally developed platforms that multiple services use for a long period. At a time of the study, Nokia had several application platform concepts. Several research projects contributed to MITA (Mobile Internet Technical Architecture), Mobile Platforms unit delivered platforms to mobile phones, and Nokia Networks had worked with for example DX200, NMS, NEMU, Flexi- and TSP platforms.

The iSource portal, meant to support collaborative development, was piloted among research projects and promoted company-wide. A corporation-wide iSource -service was established in 2003 by the Nokia IT department to support infrastructure and to promote the portal tool. A service level agreement was made between the IT department and the business units. Creation of the iSource service adds the third organization group which we use in our analyses in addition to the perspective of business unit and individual developer.

iSource is a corporation wide source code portal that for agile, fast cycle, multi-site software development (Lindman et al., 2008). The main idea behind iSource was to provide a portal enabling visibility of software and the source code inside

the company. The goals were to increase individual engineers' awareness of software developed inside the company, and to boost innovation by avoiding the problem of re-implementing the wheel. iSource's origins are in the free version of SourceForge that has been later upgraded to GForge. The web portal integrates a set of tools for use by projects including version control tools (Subversion, CVS), issue tracker, mailing lists (Mailman), forums, and file management. Today both Nokia and NSN have their own corporation wide instances of iSource.

The adoption of iSource can be divided to two phases: "bottom-up" adoption (2001-2006) and "top-down" introduction (2007-). These phases also reflect the need for portal tools, the maturity of the environment, and the company's trust on open source software.

First adopters of the portal have been leading edge research projects that were co-working with universities and research institutes. iSource has been easy to take into use in small projects, especially if co-workers were using the same tools. The iSource service released projects from the need to manage their own tools and infrastructure. The portal also provided a controlled way to work with external parties. Several projects that were first developed inside a company were open sourced later (e.g. Maemo and Python for S60).

Since the joint merger of Nokia and Siemens (2007), the focus of the service has been on launching Subversion for company wide use. During the "top down" phase the iSource portal was deployed for traditional software development driven by cost-optimization and simplification needs. Business units started to make their decisions to transition to iSource from more complex and expensive commercial tools.

Analysis, findings and discussion

On examining the cases in our study, it seems that OSS *technology* has become institutionalized in both organizations, even if detached from the classical style of developing OSS as an open endeavour. New tools have gained acceptance, provided inspiration and become familiar for the developers. Both case companies use OSS tools and processes as a way to promote software projects inside the organization.

The meaning of the term "OSS" is re-negotiated locally

In retrospect we can see a process of implicitly renegotiating the meaning of the term "OSS" to suit the organizational context. The adopted practices do not resemble OSS as understood by the "classical OSS movement", which was based on voluntarism, peer-recognition and public discussion. Instead, the OSS technology institutionalized in these two cases supports designated projects based on employment contracts. Costs are made visible and cost sharing between units is based on agreements between units. The differences are summarized in table 1.

TABLE1: Renegotiating the term "OSS".

	Classical "OSS"	Renegotiated "OSS" both at NSN and PH
Reward structure	Mostly voluntary task assignment, peer-recognition, occasionally sponsored development.	Designated projects, contributions based on employment contracts and task assignment, development costs shared based on negotiation between actors if at all.
Decision-making structure	Meritocracy, loose command structures, debates sometimes leading to crises; developers walking away from poorly functioning projects and contributing to the more attractive ones.	Hierarchical, traditional corporate chain of command, partly based on technical expertise. Some signs of seeking more consensus, though. Resources assigned to projects in project/matrix organisations.
Communication structure	Open discussion email-lists, open message boards, web-presence of projects, open documentation, open training materials. Email and instant messaging.	Intranet, visibility to selected partners who share the development costs. Use of modern *de facto* corporate communication tools such as email, instant messaging, voice calls, video conferencing etc. Some constrains due to not all information being public.

As the table above summarizes, the reward and decision-making structures are quite different, whereas the communication structure remains largely the same, when we compare the two cases to pure-form OSS projects.

Practical Tip. How "OSS" is renegotiated locally emphasizes how important it is to reserve enough time to go through the change related to the local practices in any innovation technology. The process of learning related to the new technical infrastructure and in the way of working is likely to take some time and organizational effort.

In one of the two case companies, promotion of OSS technologies was a way to sell the organizational innovation—the inner source approach—to the affected parties by aligning the change process to fit the agendas, and to serve the interests of three key groups: the business units, the central unit and developers. As a result, the changes needed for the new software development processes seem to have been easier. Despite this some groups are interested only in the tools *per se* and ignore the opportunity to share components on the inner source platform. One of the interviewees suspected that the main reason for such reluctance to share the results is in the traditional project resourcing: if a group's task is, and its success is measured by, the delivery of projects in a given time, budget and scope, then this gives no time or money to maintain or support the components in the library. Once the component projects have already been finished, the resources will have been moved on to new projects and support is no longer available from the developers most familiar with the component.

In the other case company, the promotion of the inner source approach was done more explicitly as a process change: a rationale for enabling easier reuse. Along with this process change came the technologies that are now *de facto* standard corporate tools (such as *SVN* as the version control tool). Their challenges have been on a higher level as the organization has grown through acquisitions and thus the development practices have been quite heterogeneous to start with.

The market vs. library metaphors

The inner source approaches were specifically geared towards enhancing reuse, but they present the management with an incentive issue: basically, why would a business unit contribute its developments to the inner source platform?[3]

We saw that bundling attractive tools to the platform is a way to sell the proposition of sharing. Nevertheless, the issue of support and maintenance remains—

3 In the classic, pure-form OSS development the motivational factors are quite well-known, including fun and enjoyment, peer recognition and so on, but these do not directly transfer into the corporate setting where business unit leaders make such decisions.

what's in it for the contributing group? We identified the metaphors of a private market and a local library to highlight two very different ways in which these technologies become institutionalized.

In a private market, the internal units can place their components on sale in the inner source system, and see who, if anyone is willing to buy the component at the given terms addressing use, support and maintenance. Unlike in a public market, we'll assume that in a private market there is no (or at least much less) fraud, and therefore the components can be posted openly for anyone to view, inspect and try out, but as soon as the component ends up in another group's product, this will have an internal accounting implication as per the terms and conditions agreed between the buyer and seller units. This can solve the basic incentive issue, but still leaves the resourcing issue with support and maintenance: typically a contributing unit would move on in its product development and the resources previously allocated to a given component will be reallocated to another project and other components, not allowing much time to be spent on support and maintenance of the old components. However, the currently prevailing model is still far away from a marketplace and closer to a local library model. The old component tax model is still effectively in use and brings in a price element from the market metaphor, since at least the heavy users need to pay more.

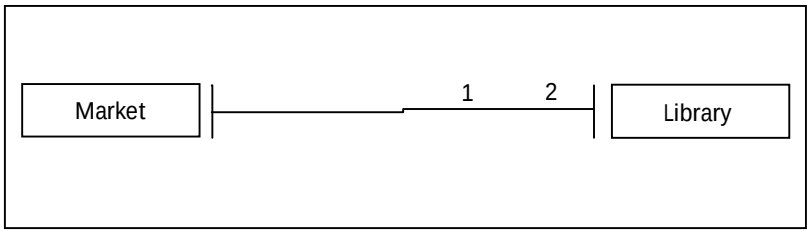
The practical difficulties of adopting such a model aside, if a particular group's components are in such high demand that others are willing to buy them at a premium, seen from the perspective of overall efficiency it would make sense for this group to focus on maintaining those components instead of starting new projects. Additionally, in hopes of more revenue, units would be promoting their components and their development on the intranet (if not globally and for all on SourceForge, for example) already before they are finalized, and thus one could expect the search costs of the users to be lowered.

The library metaphor is closer to the classic OSS licensing model: use of components is free; someone just needs to develop and contribute the components to the library. In a corporate development hierarchy, one can find platform units that get their yearly budget irrespective of the *actual* and immediate use of their components in the library. This obviously does away with the time and effort needed to negotiate between the contributor and user, but the main issue is now in central decision making: How much should be budgeted to what kind of development, and who are the people that will get the budget to perform the job? And who should make that decision?

Perhaps we should view the private market arrangement as a promising one for highly differentiating and value-adding components, whereas "corporate

commodity" components could be freely distributed in a library without complicated negotiations. If the market and library metaphors are seen as extremes of a continuum, then the two cases could be placed on that continuum roughly as follows. (1=PH, 2=NSN).

FIGURE 1: Relative positions on the continuum between a market and a library.



The private market metaphor is an appealing one—although it is in contradiction with the classical OSS spirit—and it is not surprising that in the other case company this was seriously considered. After all, it does present some benefits of open innovation (ideas flowing freely, quick diffusion of inventions to enable incremental innovation, reuse) while addressing the appropriation in a fairly practical manner.

Conclusions

In this chapter we have identified and described different ways in which OSS development practices can become institutionalised in a commercial organization. The literature emphasizes the changes brought by OSS-style development when compared to traditional development approaches in hierarchical organisations, but our data suggests that the introduction of OSS technologies and development practices has changed the two case organisations surprisingly little. However, the meaning of the term OSS has undergone considerable changes. As to how resources should be allocated inside organisations, we identified the metaphors of private markets and local libraries. Our respondents explicitly used both these metaphors when they made sense of the organisational change.

These two development organisations are embracing OSS technology in a way suitable for them: more tools, components and terminology are being adopted little by little, but the basic mode of operations still remains the same. There is no radical shift to the OSS mindset, but a slow one towards a more open and

collaborative working style, coinciding with more open communication (and, simply, more communication) and a more democratic, consensus-seeking decision making. Rather than claiming that OSS as such or OSS technologies would have changed everything in the organisational ways these corporations do software development, we'd argue that the same technological and societal developments that have contributed to the proliferation of OSS are now becoming institutionalized in hierarchical businesses.

The organisational inertia—most notably that resulting from the way budgeting and project management are performed within a large development organization—can be used to explain how large development organisations mould and redefine "OSS" to fit their old trajectory. It seems that companies have considerable leeway and interpretive flexibility in determining what their processes are like even if they were labelled as open.

References

Dahlander, L. and Magnusson, M. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy 34*, 481-493.

DiMAggio, P and Powell, W. (1991). Introduction. In Powel, W. and Dimaggio (Eds.) *The new institutionalism in organizational analysis*. 1-38. Chicago, University of Chicago Press.

Fink, M. (2003). *The Business and Economics of Linux and Open Source*. Upper Saddle River, NJ: Prentice Hall PTR.

Fitzgerald, B. (2006). The Transformation of Open Source Software. *MIS Quarterly 30*(3), 587-598.

Garud, R., Hardy, C., and Maguire, S. (2007). *Organization Studies 28*, 957-969.

Garud, R., and P. Karnøe. (2003). Bricolage vs. breakthrough: Distributed and embedded agency in technology entrepreneurship. *Research Policy 32*, 277–300.Greenwood R., and Hinings, C.R. (1996). Understanding radical organizational change: bringing together the old and the new institutionalism. *Academy of Management Review 21*(4), 1022-1054.

Klein, H. K. and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Quarterly 23* (1), 67-94.

Powell, W.W., and DiMaggio P.J. (Eds.) (1991). The new institutionalism in organisational analysis. Chicago: University of Chicago Press.

Rajala, R., Nissilä, J. and Westerlund,M. (2006). Determinants of Open Source Software Revenue Model Choices. Proceedings of the 14th European Conference on Information Systems (ECIS 2006), 12 - 14 June, Göteborg, Sweden.

Seaman, C. B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering 25* (4), 557-572.

Scott, W.R. (2001). *Institutions and Organizations*, 2nd ed.. CA, Thousand Oaks.

Stol, K. and Babar, M. (2009). Reporting empirical research in open source software: the state of practice, in Boldyreff, C., Crowston, K., Lundell, B., Wasserman, A. (eds.) Proceedigns of the 5th Conference on Open Source Ecosystems: Diverse Communities Interacting, June 3rd-

6th, Skövde, Sweden, IFIP Advances in Information and Communication Technology 299/2009, Springer 2009, 156-169.
Swanson, B., and Ramiller, N. (1997). The Organizing Vision in Information Systems Innovation. *Organization Science 8* (5), 458-474.
Van de Ven, A.H. (1993). Managing the Process of Organizational Innovation in Huber, G.P. and Glick, W.H. (Eds.). Organizational Change and Redesign: Ideas and Insights for Improving Performance. Oxford University Press, New York.
Wesselius, J. (2008). The Bazaar inside the Cathedral: Business Models for Internal Markets. *IEEE Software 25* (3), 60-66.

Further reading

Barnett, L. (2004). Applying Open Source Processes in Corporate Development Organisations. (http://www.forrester.com/rb/Research/applying_open_source_processes_in_corporate_development/q/id/34466/t/2, Forrester Research.
Dinkelacker, J., Garg, P., Miller, R. and Nelson, D. Progressive open source. In Proceedings of ICSE 2002, 177-184.
Gurbani V., Garvert, A., and Hersleb, J. (2010). Managing a Corporate Open Source Asset. *Communications of the ACM 53 (*2), 155-159.
van der Linden, F., Lundell, B., Marttiin, P. (2009). Commodification of Industrial Software - a Case for Open Source. *IEEE Software 26* (4),77-83.
Lindman, J., Rossi, M., and Marttiin, P. (2008). Applying Open Source Development Practices Inside a Company. 4th International Conference on Open Source Systems. 7-10 September 2008, Milan, Italy.
Santos, C. (2008). Understanding Partnerships between Corporations and the Open Source Community: A Research Gap. *IEEE Software 25(*6), 96-97.
Sharma, S., Sugumaran, V. and Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal 12*(1), 7-25.
Välimäki, M. (2005). The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry. Helsinki University of Technology, Helsinki, Finland.